A matheuristic approach for the two-machine total completion time flow shop problem

Published version:

DOI:10.1007/s10479-011-0928-x

(Article begins on next page)

Federico Della Croce; Andrea Grosso; Fabio Salassa. A matheuristic approach for the two-machine total completion time flow shop problem. ANNALS OF OPERATIONS RESEARCH. 213 pp: 67-78.
DOI: 10.1007/s10479-011-0928-x

A MATHEURISTIC APPROACH FOR THE TWO-MACHINE TOTAL COMPLETION TIME FLOW SHOP PROBLEM

FEDERICO DELLA CROCE[1], ANDREA GROSSO[2], FABIO SALASSA[1]
1. D.A.I., POLITECNICO DI TORINO, ITALY
2. DIP. INFORMATICA, UNIVERSITÀ DI TORINO, ITALY

Abstract. This paper deals with the two-machine total completion time flow shop problem. We present a so-called *matheuristic* post processing procedure that improves the objective function value with respect to the solutions provided by state of the art procedures. The proposed procedure is based on the positional completion times integer programming formulation of the problem with $O(n^2)$ variables and $O(n)$ constraints.

1. Introduction

In the present work a *matheuristic* solution approach is proposed for minimizing the total (or average) completion time in a 2-machine flow shop problem ($F2| \,|\sum C_i$ in the three-fields notation of Graham et al. [12]). In a 2-machine flow-shop environment a set of jobs $N = \{1, 2, \ldots, n\}$ is to be scheduled on two machines, and each job $i \in N$ is made up of two *operations,* the first one (respectively, the second) requiring to run continuously for p_{1i} (resp. p_{2i}) units of time on the first (resp. second) machine. For each job, the second operation cannot begin if the first one is not completed. The completion time C_i of a job $i \in N$ in a schedule S is defined as the completion time of its second operation. The $F2| \,|\sum C_i$ problem calls for finding a schedule S that minimizes

$$f(S) = \sum_{i \in N} C_i(S).$$

The problem is known to be NP-complete; also, at least an optimal solution is known to be a *permutation schedule,* where the (operations of the) jobs share the same sequence on both machines. Thus we deal equivalently with the *permutation* flow shop problem $F2|\text{perm}|\sum C_i$. The flow shop problem is one of the oldest and best known production scheduling models and the available literature is extensive. We refer to [20, 19, 9, 16] for contributions related to the objective function tackled in this work. Exact algorithms (mainly "ad-hoc" branch and bound, [7, 4, 2, 1, 15, 22]) and MILP-based approaches [21], have also been proposed, but due to their important computational times, these methods are mainly suitable to solve relatively small size instances.

This work concerns a novel heuristic approach to the $F2| \,|\sum C_i$ problem. To the authors' knowledge the best results – as far as heuristic approaches are considered — obtained for the $F2| \,|\sum C_i$ problem have been achieved by the Recovering Beam Search method (RBS), a truncated implicit enumeration enhanced by local search — see [5] for details. Also, Dong et al. [9] proposed a very effective Iterated Local Search method for the more general m-machines permutation flow-shop ($Fm|\text{perm}|\sum C_i$), but computational experience is not reported for the 2-machine case. Similarly, in [3] an hybrid, and very effective approach outperforming that of [9] has been proposed for the same m-machines problem. Also in this case, computational experience is not reported for the 2-machine case.

Matheuristics are methods that recently attracted the attention of the community of researchers, suddenly giving rise to an impressive amount of work in a few years. Matheuristics lye on the general idea of exploiting the strength of both metaheuristic algorithms and exact methods as well, leading to a "hybrid" approach (see [18]), but because of their novelty there is no unique classification nor a consolidated working framework in the field; hence, it is hard to state a pure and sharp definition of these methods.

A distinguishing feature is often the exploitation of nontrivial mathematical programming tools as part of the solution process. For example, in [10] a sophisticated Mixed-Integer Linear Programming (MILP) solver is used for analyzing very large neighborhoods in the solution space.

A crucial issue also underlined in [18], is that the structure of these methods is not a priori defined and in fact a solution approach can be built in many different ways. As a general example, one can construct a matheuristic algorithm based on an overarching well known Metaheuristic, a Variable Neighborhood Search for example [8, 13], with search phases realized by an exact algorithm as well as by a MILP solver. A different, more loosely coupled approach could be a two-stage procedure: a first heuristic procedure is applied to the problem for generating a starting solution and then a post processing "refinement" procedure is applied exploiting, for example, some peculiar properties of the mathematical formulation of the problem under analysis; this second example is the core idea of the present work.

Pursuing the above sketched idea of a two-stage procedure, we couple a heuristic algorithms like RBS with a neighborhood search based on a MILP formulation solved by means of a commercial tool. The two-stage approach is appealing because of its simplicity — allowing to tinker with building blocks plus some glue-code — and for the possibility of concentrating more on modeling the neighborhood instead of building up the search procedure. Exploiting this idea we get very good results, improving solution's quality over the state of the art heuristics.

The paper is organized as follows: in Section 2 a MILP model for the problem is recalled and the proposed matheuristic procedure is described. In Section 3 computational results are reported, and final remarks are given in Section 4.

A preliminary version of the discussed results has been presented at the EvoCop 2011 conference [6].

2. Basic model and matheuristic approach

Following the two-stage scheme we execute in the second stage an intensive neighborhood search starting from the solution delivered in the first stage. We define a neighborhood structure relying on a MILP model of the $F2|\text{perm}|\sum C_j$ problem; we stand on the model with positional variables (see [17, 14]), since it offered better performances with respect to other classical models based on disjunctive variables and constraints.

Let C_{ki} be variables representing the completion times of i-th job processed by machine $k = 1, 2$ and x_{ij} 0/1 decision variables, where $i, j \in \{1, \ldots, n\}$. A variable x_{ij} is equal to 1 if job i is in position j of the sequence, zero otherwise.

The problem can be formulated as follows.

$$\min \sum_{j=1}^{n} C_{2j} \tag{1}$$

subject to

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad \forall j = 1,\dots,n \tag{2}$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad \forall i = 1,\dots,n \tag{3}$$

$$C_{11} = \sum_{i=1}^{n} p_{1i} x_{i1} \tag{4}$$

$$C_{21} = C_{11} + \sum_{i=1}^{n} p_{2i} x_{i1} \tag{5}$$

$$C_{1j} = C_{1,j-1} + \sum_{i=1}^{n} p_{1i} x_{ij} \qquad \forall j = 2,\dots,n \tag{6}$$

$$C_{2j} \geq C_{1j} + \sum_{i=1}^{n} p_{2i} x_{ij} \qquad \forall j = 2,\dots,n \tag{7}$$

$$C_{2j} \geq C_{2,j-1} + \sum_{i=1}^{n} p_{2i} x_{ij} \qquad \forall j = 2,\dots,n \tag{8}$$

$$x_{ij} \in \{0,1\} \tag{9}$$

where constraints (2)–(3) state that a job is chosen for each position in the sequence and each job is processed exactly once. Constraints (4)–(6) set the completion time of the first job on both machines. Constraints (7)–(8) forbid for each job the start of the 2-nd operation on the corresponding machine *two* before its preceding operation on machine *one* has completed.

The heuristic algorithm considered for the first stage is RBS from [5]; RBS is a beam search technique combined with a limited neighborhood search typically based on job extraction and reinsertion. For the $F2|\ |\sum C_i$ problem it offers high execution speed combined with a good solution quality.

In designing a neighborhood concept for the second-stage search, a crucial issue is that the structure of the neighborhood should be as much as possible "orthogonal" to the structure of the neighborhoods used by the first-stage heuristic. That is, we do not want the solution delivered by the first stage to be (close to) a local optimum for the second stage. Hence we tried to design a neighborhood with many more degrees of freedom, still keeping in mind that the perturbation of the current solution should not fully disrupt its structure.

Consider a working sequence $\bar{S}$; in model (1)–(9) this obviously corresponds to a valid configuration $\bar{x} = (\bar{x}_{ij} : i,j = 1,\dots,n)$ satisfying constraints (2)–(3), with $\bar{x}_{ij} = 1$ iff job i appears in the j-th position of $\bar{S}$. We define a neighborhood $\mathcal{N}(\bar{S}, r, h)$ by choosing a position r in the sequence and a "size" parameter h; let $\bar{S}(r;h) = \{[r],[r+1],\dots,[r+h-1]\}$ be the index set of the jobs located in the consecutive positions $r,\dots,r+h-1$ of sequence $\bar{S}$ — we call such run a "job-window". The choice of the best solution in the neighborhood $\mathcal{N}(\bar{S}, r, h)$ is accomplished by <u>minimizing</u> (1) subject to (2)–(9) *and*

$$x_{ij} = \bar{x}_{ij} \qquad \forall\ i \notin \bar{S}(r;h),\ j \notin \{r,\dots,r+h-1\}. \tag{W}$$

The resulting minimization program — we call it the *window reoptimization* problem — is solved by means of an off-the-shelf MILP solver. The additional constraints (W) state that in the new solution all jobs but those in the window are fixed in the position they have in the current solution, while the window gets reoptimized — the idea is sketched in Figure 1.

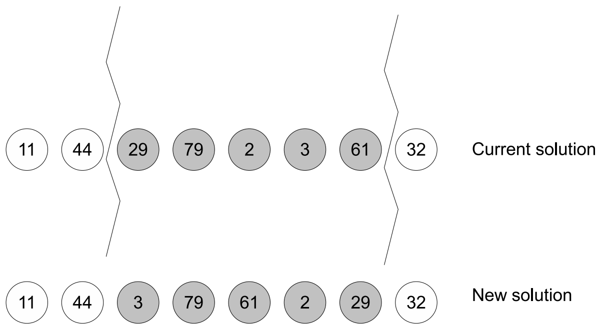

Figure 1. Example of jobs window reoptimized

If no improved solution is found a new job-window is selected to be optimized until all possible $O(n)$ windows have been selected. The search is stopped because of local optimality (no window reoptimization offers any improved solution) or because a predefined time limit expires.

It is known that exact methods are usually not suited for this kind of problems because of the amount of CPU time they need to solve a problem but this is true for larger size problems while they can perform well only on relatively small size instances. Exploiting this issue with our approach, a subproblem is generated with few variables and in such case we know that commercial, open source or custom exact methods can be well performing at analyzing large scale (exponential) neighborhoods of a given solution.

With respect to the choice of the windows, a first-improvement strategy has been implemented: as soon as an improved solution is found, solving a window reoptimization problem, that solution becomes the new current. The choice of the windows (the r index) is randomized — keeping track of the already examined windows. The algorithm can be schematically described as follows.

$\bar{x} = \langle$heuristic solution from 1st stage$\rangle$
repeat
  Set improved := false;
  repeat
    Pick $r \in \{1, \ldots, n-h+1\}$ randomly;
    Compute $\bar{S}(r; h)$;
    minimize (1) subject to (2)–(9) and (W)
    Let $\hat{x}$ be the optimal solution;
    if $f(\bar{x}) > f(\hat{x})$ **then**
      $\bar{x} := \hat{x}$
      Set improved := true;
    end if
  until improved **or** all r values have been tried
until not improved **or** time limit expired

In order to limit the time to search a window we stop the window reoptimization after a time limit T_w, concluding with the best incumbent available and the neighborhood being only partially searched. We note anyway that for reasonable values of T_w most of the times the window reoptimization can be fully performed.

Our design choices rest on top of a preliminary computational study, as we outline below.

Tests performed in order to compare the performances of models based on disjunctive constraints against models with positional variables pointed out that window reoptimization becomes substantially less efficient, requiring higher computation time. This phenomenon was quite expected since disjunctive models are popularly considered weaker than positional models because of the substantial number of "big-M" constraints involved in such formulations.

Decision taken about generating neighbors based on windows is justified by preliminary tests conducted on a pool of instances of various sizes. In principle, an even simpler neighborhood definition could require the reoptimization of a completely general subset of jobs — not necessarily consecutively sequenced. Anyway it turned out that often the first-stage solutions delivered by RBS were nearly local minima for neighborhoods based on rescheduling non consecutive jobs, thus missing the desired "orthogonality" between first and second-stage neighborhoods. This phenomenon is much less common when using windows.

The window size h is the key parameter in our approach; its choice is dictated by the need of trading off between the chances of improving a given solution and the CPU time the solver needs to actually perform the reoptimization. A small window size makes reoptimization faster, but of course it restricts the size of the neighborhood; on the other hand the neighborhood should be, obviously, as large as possible in order to have more chances of improving the current solution. After testing the same pool of instances, we fixed $h = 12$; this value proved to be a robust choice, giving good results through extensive computational tests (see Section 3) on instances with 100, 300 and 500 jobs.

The value $h = 12$ should be considered only as an indication of the order of magnitude for the parameter: note that the choice may also depend on the technology of the underlying solver — that is used in a "black-box" fashion, and whose internals may not be fully known.

3. Computational Results

We ran tests on a Xeon processor at 2.33 GHz, with 8 GB of RAM; CPLEX 12.1 was used as MILP solver. CPLEX default parameters were kept, without attempting to tune them. In order to generate the first stage solution of each instance we ran RBS (from [5]) with beam size 10. Computational experience showed that widening such parameter does not significantly improve the performances of RBS.

Tables 1–3 report the performances achieved by the two-stage procedure on instances generated as in [5], for $n = 100, 300, 500$, with integer processing times randomly drawn from the uniform distribution $[1, 100]$. The first stage (RBS) never consumed more than 0.4, 4 and 15 seconds of CPU time respectively and we allowed the second stage to run with a time limit of 60, 600, and 3600 seconds respectively for the three problem sizes. The time limit T_w for the window reoptimization was set to 10, 60 and 100 seconds for $n = 100, 300, 500$ respectively, but in all cases the window reoptimization was achieved well before the limit. The tables compare the objective value of solutions delivered by the matheuristic approach against those delivered by pure RBS, ILS [9] and SAwGE [3].

Both the latter two algorithms use a neighborhood definition based on job swap and job extraction/reinsertion; ILS executes repeated neighborhood searches, and restarts the search by randomly perturbing the best known solution after a prefixed number of non-improving searches have been performed. SAwGE runs a "population" of simulated annealing local searches with different parameters settings, replacing such settings after they have been failed to improve the best known solution for a given number of attempts; the algorithm is naturally designed for a parallel computing environment, but runs on a single processor as well.

ILS and SAwGE ran for the same time limit given to the two-stage procedure; the ILS code was kindly provided by the authors of [9] as well as results of SAwGE, on all our test instances, were kindly provided by the author of [3]. We note that the machine used for SAwGE (Intel i7 980X 3.33 GHz) can be estimated to be approximately 40% faster than our processor. Also, the upper and lower bounds provided by CPLEX branch and cut running on model (1)–(9) within the same allowed time limits of 60, 600 and 3600 seconds are reported — the LB column gives the minimum lower bound of the remaining open nodes.

Table 1 is related to the tests for $n = 100$. We note that CPLEX delivered, after one minute of search, a solution whose quality is in most cases dominated by that of ILS, RBS and SAwGE — hence we do not consider "pure" CPLEX a strong competitor, and focus with the comparison with SAwGE, which is the strongest one. Instead, we remark that running the second-stage search for the same amount of time allows a more effective use of the solver. Columns 2 reports the result of the first stage (namely RBS with beam size 10). Column 3 reports the results of ILS within the time limit of 60 seconds. Columns 4 and 5 depict the average results on 5 runs of the SAwGE approach and our matheuristic algorithm within the same time limit of 60 seconds, while column 6 (MATHEUR*) reports the results of our approach within a time limit of 300 seconds. The reason to test our procedure against wider time limits is justified by the wish to verify if a local minimum has been reached in the benchmark time limit or if the time limit stopped the approach while improvements were still to be found. The next three columns report the results of the best values among the 5 runs. Finally CPLEX LB and UB are reported. For the case of $n = 100$ all heuristics allow for a narrow optimality gap, in all cases but one the two-stage search strongly dominates all the other compared approaches and if a larger time limit is considered, our approach is better in all cases depicted with the *italic* character — this asserts the effectiveness of the second-stage neighborhood, which offers large margins of improvements, although these must be traded-off with CPU time. Moreover, only for four instances over twenty, 60 seconds were sufficient to find a local minimum.

Same considerations can be replicated for Table 2 were we consider $n = 300$. In this case the benchmark time limit is 600 seconds while the extended time limit is 1800 seconds. The matheuristic approach gave better results in all cases but two while within the extended time limit our procedure was always better. Moreover, 600 seconds were never sufficient to certify a local minimum.

Results presented in Table 3 for $n = 500$ jobs were less effective than the other tests. In this case, in fact, SAwGE performed better than our procedure on 13 instances over 20 within the time limit of 1 hour. Considering the extended time limit of 2 hours our approach resulted to be better than SAwGE 12 times which confirmed us that we were still far from a local minima for our procedure.

Although the comparison is done with time limits of 60, 600 and 3600 seconds we note that, when MATHEUR is the winner, a solution with value *below* the average value delivered by the main competitor SAwGE is found by MATHEUR in a considerably shorter time: this happens on average after 16, 187, 1606 seconds respectively for the tests with $n = 100, 300, 500$ jobs.

Finally, we also tested the second-stage search within the time limit of one hour on the solutions delivered by SAwGE for $n = 500$; the results are reported in Table 4. Interestingly, for all instances, the solutions of SAwGE were not local minima for our post-processing refinement, which confirmed us that the second-stage neighborhood is also orthogonal to the SAwGE neghborhoods and can be robustly cascaded to different metaheuristics or hybrid heuristics in order to improve their performances.

Relying on the results achieved, the proposed approach can be claimed to strongly outperform most of the state of the art heuristics on medium and large-sized instances. On the very large 500-jobs instances the approach is strongly competitive with SAwGE, while leaving room for improvement since the delivered solutions within the allowed time limits are far from local optima. Performances on the largest instances could be improved by further calibration of the window size — only a mild effort has been devoted to it in this work — and/or incorporating some diversification technique in the second-stage search, that up to now consists of pure intensification.

4. Concluding Remarks

A matheuristic two-stage approach for minimizing total flowtime in a (permutation) 2-machine flow shop has been developed and tested. The obtained results confirm that even if, apparently, MILP approaches still cannot compete with ad-hoc state of the art heuristics for such problem, an hybrid and *simple* approach implementing a post-optimization refinement procedure by means of a MILP solver can achieve valuable results, dominating the current state-of-the art heuristics. Preliminary tests on the more general $Fm|\text{perm}|\sum C_i$ problem showed that the proposed two-stage approach was apparently less successful at least with respect to results and CPU times provided in [3]. This is probably due to the fact that the gap between the ILP model solution value and the ILP model continuous relaxation solution value increases as the number of machines increases inducing a much larger effort for computing each neighbor in the second stage of our approach. Future research will be devoted to incorporate diversification techniques and to better calibrate the window size in order to improve the performances on the $Fm|\text{perm}|\sum C_i$ problem.

References

[1] Akkan C., Karabati S., The two-machine flowshop total completion time problem: Improved lower bounds and a branch-and-bound algorithm. European Journal of Operational Research, 159, pp. 420-429, 2004.

[2] Bansal S.P., Minimizing the sum of completion times of n-jobs over M-machines in a flowshop. A branch and bound approach. AIIE Transactions, 9, pp. 306-311, 1977.

[3] Czapiński M., Parallel Simulated Annealing with Genetic Enhancement for flowshop problem with C_{sum}. Computers & Industrial Engineering, 59, pp. 778-785, 2010.

[4] Della Croce F., Ghirardi M., Tadei R., An improved branch-and-bound algorithm for the two machine total completion time flow shop problem. European Journal of Operational Research, 139, pp. 293-301, 2002.

[5] Della Croce F., Ghirardi M., Tadei R., Recovering Beam Search: enhancing the beam search approach for combinatorial optimization problems. Journal of Heuristics, 10, pp. 89-104, 2004.

[6] Della Croce F., Grosso A., Salassa F., A Matheuristic Approach for the Total Completion Time Two-Machines Permutation Flow Shop Problem Lecture Notes in Computer Science, 6622, pp.38-47, 2011 (*Proceedings of EvoCop 2011, Torino, Italy, April 27-29, 2011*).

[7] Della Croce F., Narayan V., Tadei R., The two-machine total completion time flow shop problem. European Journal of Operational Research, 90, pp. 227-237, 1996.

[8] Della Croce F., Salassa F., A Variable Neighborhood Search Based Matheuristic for Nurse Rostering Problems, PATAT 2010 Proceedings, 8th International Conference on the Practice and Theory of Automated Timetabling, Belfast (UK) 10 - 13 August 2010, pp. 167-175.

[9] Dong X., Huang H., Chen P., An iterated local search algorithm for the permutation flowshop problem with total flowtime criterion. Computers & Operations Research, 36, pp. 1664-1669, 2009.

[10] Fischetti M., Lodi A., Local Branching. Mathematical Programming B, 98, pp. 23-47, 2003.

[11] Garey M.R., Johnson D.S., Sethi R., The complexity of flowshop and jobshop scheduling, Mathematics of Operations Research, 1, pp. 117-129, 1976.

[12] Graham R. L., Lawler E. L., Lenstra J. K., Rinnooy Kan A. H. G., Optimization and approximation in deterministic sequencing and scheduling: a survey, Annals of Operations Research, 5, 287-326, 1979.

[13] Hansen P., Mladenovic N., Variable neighborhood search: Principles and applications, European Journal of Operational Research, 130, pp. 449-467, 2001.

[14] Hoogeveen H., van Norden L., van de Velde S., Lower bounds for minimizing total completion time in a two-machine flow shop. Journal of Scheduling, 9, pp. 559-568, 2006

[15] Hoogeveen J. A., Van de Velde S. L., Stronger Lagrangian bounds by use of slack variables: applications to machine scheduling problems. Mathematical Programming, 70, pp. 173190, 1995.

[16] Ladhari T., Rakrouki M.A., Heuristics and lower bounds for minimizing the total completion time in a two-machine flowshop. International Journal of Production Economics, 122, pp. 678-691, 2009.

[17] Lasserre J.B., Queyranne M., Generic scheduling polyhedral and a new mixed integer formulation for single machine scheduling. in Proceedings of the IPCO Conference, 1992, pp. 136-149.

[18] Maniezzo V., Stutzle T., Voss S., Matheuristics: Hybridizing Metaheuristics and Mathematical Programming, Annals of Information Systems, 10, Springer,2009.

[19] Ruiz R., Maroto C., A comprehensive review and evaluation of permutation flowshop heuristics. European Journal of Operational Research, 165, pp. 479-494, 2005.

[20] Taillard E., Some efficient heuristic methods for the flow shop sequencing problem. European Journal of Operational Research, 47, pp. 65–74, 1990.

[21] Stafford E.F., On the development of a mixed integer linear programming model for the flowshop sequencing problem. Journal of the Operational Research Society, 39, pp. 1163-1174, 1988.

[22] Van de Velde S., Minimizing the sum of job completion times in the two-machine flowshop by Lagrangean relaxation, Annals of Operations Research, 26, pp. 257-268, 1990.

Inst	RBS	ILS	avg SAwGE	avg MATHEUR	avg MATHEUR*	best SAwGE	best MATHEUR	best MATHEUR*	CPLEX LB	CPLEX UB
0	190580	190599	190435.60	**190240.80**	*190230.40*	190299	**190233**	*190223*	189843.64	190637
1	192250	192346	192310.00	**192157.00**	*192157.00*	192217	**192155**	*192155*	191741.95	192541
2	175496	175586	175510.30	**175336.00**	*175326.60*	175374	**175322**	*175321*	174909.20	175694
3	190894	190822	190735.60	**190544.80**	*190541.80*	190636	**190539**	*190539*	190100.11	191192
4	173260	172787	172729.20	**172647.20**	*172607.60*	172658	**172598**	*172597*	171923.31	173041
5	187234	187226	187079.60	**186952.60**	*186942.80*	186991	**186942**	*186942*	186623.70	187228
6	166930	166154	166075.90	**166002.80**	*165952.20*	166010	**165956**	*165951*	165449.77	166294
7	193122	193106	193037.60	**192893.40**	*192892.40*	192940	**192881**	*192876*	192529.62	193112
8	171435	171374	171317.90	**171229.80**	*171193.00*	171232	**171186**	*171182*	170762.77	171575
9	173666	173470	173465.70	**173354.60**	*173351.40*	173351	**173349**	*173345*	172822.78	173619
10	194496	194164	194123.40	**194063.80**	*194028.20*	194062	**194035**	*194013*	193602.12	194378
11	178703	178713	178555.70	**178410.60**	*178403.00*	178435	**178403**	*178399*	178016.53	178747
12	188090	187590	187327.90	**187238.20**	*187190.00*	187214	**187193**	*187183*	186786.80	187516
13	204725	205101	204569.60	**204422.20**	*204400.60*	204431	**204408**	*204389*	203950.38	204856
14	186049	186173	185990.80	**185688.40**	*185647.00*	185726	**185657**	*185643*	185166.21	186182
15	194925	194619	194600.40	**194454.20**	*194382.00*	194493	**194404**	*194379*	193773.17	194977
16	192079	192382	192041.30	**191910.00**	*191851.20*	191947	**191887**	*191844*	191282.61	192466
17	202725	202021	**201939.40**	201946.40	*201811.20*	201873	**201872**	*201800*	201291.25	202273
18	176443	176216	176193.80	**176141.20**	*176094.00*	**176121**	176133	*176067*	175570.47	176514
19	178313	178349	178179.10	**178027.00**	*178000.00*	178035	**178007**	*177993*	177651.31	178210

TABLE 1. Results for $n = 100$, 60 secs.

Inst	RBS	ILS	avg			best			CPLEX	
			SAwGE	MATHEUR	MATHEUR*	SAwGE	MATHEUR	MATHEUR*	LB	UB
0	1703211	1702256	1701189.80	**1699970.00**	*1699468.60*	1700685	**1699751**	*1699454*	1697849.40	1702034
1	1632209	1636048	1632173.40	**1630957.00**	*1630770.20*	1631645	**1630848**	*1630670*	1629384.60	1632681
2	1737151	1738890	1737303.80	**1736185.80**	*1736015.60*	1736530	**1736168**	*1735984*	1733880.94	1738066
3	1761128	1762538	1760404.20	**1759338.00**	*1759189.20*	1760156	**1759249**	*1759112*	1757575.31	1761260
4	1836372	1840062	1835444.60	**1834489.00**	*1834192.20*	1834984	**1834379**	*1834149*	1832929.82	1836067
5	1765979	1768072	1766194.00	**1764612.40**	*1764347.60*	1765694	**1764582**	*1764305*	1762980.85	1766312
6	1797299	1798523	1795125.80	**1794258.40**	*1793320.00*	1794256	**1794095**	*1793281*	1791780.67	1795096
7	1803321	1806213	1803265.20	**1802117.60**	*1801975.60*	1802635	**1802080**	*1801956*	1800512.03	1804189
8	1784953	1780052	**1777283.20**	1778835.60	*1776158.40*	**1776529**	1778030	*1776070*	1773943.34	1778085
9	1835515	1840146	1833128.60	**1832611.60**	*1832001.80*	1832643	**1832528**	*1831972*	1830603.61	1833678
10	1739093	1740728	**1736751.00**	1736758.80	*1736001.80*	1736622	**1736557**	*1735947*	1734497.71	1737675
11	1755873	1749832	**1747343.60**	1748477.00	*1746385.40*	**1747097**	1747779	*1746321*	1744370.65	1748522
12	1820591	1823739	1818196.80	**1817563.00**	*1816850.80*	1817887	**1817335**	*1816771*	1815519.06	1818330
13	1691438	1692350	1689676.40	**1688847.60**	*1688307.60*	1688929	**1688663**	*1688054*	1686405.61	1690935
14	1792010	1795283	1790632.20	**1789800.20**	*1789424.20*	1790436	**1789677**	*1789360*	1788031.32	1791391
15	1830889	1832292	1830755.20	**1829696.00**	*1829535.60*	1830246	**1829637**	*1829460*	1828170.46	1832264
16	1825747	1826510	1822464.00	**1822093.40**	*1821484.60*	1822101	**1821857**	*1821391*	1819986.79	1823464
17	1756024	1757753	1755691.20	**1753966.20**	*1753652.40*	1754807	**1753726**	*1753579*	1752198.10	1756183
18	1756944	1758224	1755593.60	**1754664.40**	*1754354.80*	1755129	**1754426**	*1754310*	1752940.80	1756736
19	1827704	1828647	1826968.00	**1825588.80**	*1825370.60*	1826542	**1825512**	*1825351*	1823857.26	1828020

TABLE 2. Results for $n = 300$, 600 secs.

Inst	RBS	ILS	avg SAwGE	avg MATHEUR	avg MATHEUR*	best SAwGE	best MATHEUR	best MATHEUR*	CPLEX LB	CPLEX UB
0	5144540	5152875	5142663.00	**5141035.00**	*5140088.00*	5141524	**5140704**	*5139879*	5136952.93	5143635
1	4994590	5002745	**4988122.00**	4989330.60	*4987199.00*	**4986659**	4988718	4986863	4983253.08	4988865
2	5040922	5034092	**5026592.40**	5031956.00	5028282.20	**5026005**	5031439	5027859	5020723.19	5029116
3	4904534	4914064	4902607.40	**4901596.00**	*4900553.00*	4901829	**4901390**	*4900397*	4897858.55	4904571
4	5005902	5009590	5003460.40	**5002645.00**	*5001495.60*	**5001995**	5002321	*5001208*	4997890.36	5004911
5	4842945	4853143	**4838073.40**	4838429.00	*4837207.60*	**4837547**	4838034	*4836893*	4833972.24	4839728
6	5011513	5016199	5007889.60	**5006797.80**	*5005706.40*	5006636	**5006543**	*5005414*	5002655.02	5009555
7	5104328	5076191	**5071293.40**	5089775.20	5081800.80	**5069880**	5083798	5076701	5064833.20	5071688
8	5063940	5065326	5062120.60	**5059422.20**	*5058850.80*	5061110	**5059101**	*5058743*	5056099.72	5062880
9	4838541	4846072	4838477.00	**4836007.60**	*4835220.80*	4836332	**4835829**	*4835029*	4832246.66	4839079
10	5092817	5097629	**5088385.20**	5088913.20	*5087381.60*	**5087497**	5088753	*5087221*	5083138.52	5090212
11	4772735	4766853	**4760660.80**	4763886.00	*4760604.80*	**4759532**	4762728	4759942	4754100.99	4761279
12	4908764	4922216	4906880.00	**4905959.60**	*4905085.60*	4906437	**4905598**	*4904884*	4902325.86	4911806
13	4814208	4808437	**4794109.80**	4803594.80	4797696.60	**4793309**	4802559	4797126	4789764.69	4795061
14	5038041	5042309	5036140.60	**5034942.60**	*5033956.20*	**5033911**	5034559	*5033664*	5029369.95	5038432
15	5285359	5265594	**5259570.60**	5273075.20	5266745.80	**5259064**	5270753	5264235	5254046.68	5261292
16	4799251	4801812	**4792690.60**	4795317.20	4793679.20	**4791951**	4794986	4793164	4788121.47	4797003
17	4957518	4968088	4956408.00	**4955503.20**	*4954913.40*	4956028	**4955382**	*4954798*	4952415.71	4959358
18	4860580	4870398	**4852540.60**	4855075.40	*4852439.40*	**4851754**	4854128	4852002	4848152.27	4855376
19	5059799	5065701	5056535.30	**5056229.00**	*5054837.20*	**5055991**	5056070	*5054660*	5051654.98	5058377

TABLE 3. Results for $n = 500$, 3600 secs.

Inst	SAwGE	MATHEUR
0	5141524	**5140149**
1	4986659	**4985716**
2	5026005	**5024975**
3	4901829	**4900729**
4	5001995	**5000896**
5	4837547	**4836580**
6	5006636	**5005603**
7	5069880	**5068810**
8	5061110	**5059356**
9	4836332	**4835392**
10	5087497	**5086158**
11	4759532	**4758150**
12	4906437	**4905328**
13	4793309	**4792318**
14	5033911	**5032551**
15	5259064	**5257936**
16	4791951	**4790804**
17	4956028	**4955045**
18	4851754	**4850806**
19	5055991	**5054790**

TABLE 4. Improvement obtained by the 2nd stage run on SAwGE's solution.